A Color-Coded Vision Scheme for Robotics

Kelley Tina Johnson
NASA/Goddard Space Flight Center
Robotics Data Systems & Integration Section
Code 735.1
Greenbelt, MD 20771

ABSTRACT

Most vision systems for robotic applications rely entirely on the extraction of information from gray-level images. Humans, however, regularly depend on color to discriminate between objects. Therefore, the inclusion of color in a robot vision system seems a natural extension of the existing gray-level capabilities.

This paper discusses a method for robot object recognition using a color-coding classification scheme. The scheme is based on an algebraic system in which a 2-dimensional color image is represented as a polynomial of two variables. The system is then used to find the color contour of objects. In a controlled environment, such as that of the in-orbit space station, a particular class of objects can thus be quickly recognized by its color.

Keywords: Image Processing, Image Algebra, Color, Artificial Intelligence, Vision

INTRODUCTION

The continued evolution of robotics in America's space program is critical to the advancement of space science. Robots have contributed to the launching of satellites, the collection of lunar surface samples, and a wide array of other NASA activities. As the Space Station Freedom develops in the 1990's, robotics will continue to play a major role.

In response to the anticipated need for routine assembly and maintenance tasks required by the space station, NASA has outlined plans for an advanced robotic system, the FTS (Flight Telerobotic Servicer), to aid in these tasks. Initially, the FTS will assist the astronauts in building the space station. Later, it will perform routine maintenance.

Space Station servicing will require a considerable amount of work to be done outside the spacecraft. The extreme dangers and cost of astronauts performing EVA (extra-vehicular activity) prohibit a repair force from being 100% human. Robotics will necessarily have to be incorporated as a substitute for human labor. Designed correctly, the robot can perform the tasks as capably as a human, but without the same health risks.

The effectiveness of a robot in a particular situation can often be accredited to the control of its human-like appendages. Just as a human relies on arms, legs, feet, hands, etc. to accomplish tasks, robots also have limbs with which to maneuver and to work. However, what is generally overlooked when discussing robotics is the required sensory input a robot needs to remain productive in its environment.

Humans benefit from continuous interaction with sound, light, temperature, etc., but a robot must be specifically adapted to sense and to understand these same stimuli. Because humans rely on vision more than any other sensory input, vision should be a major concern for robotics researchers.

Unfortunately, duplicating human vision capabilities for a robot is not a trivial task. The massive parallel structure of the brain makes current SISD (single instruction, single data stream) digital computer technology obsolete. Even state-of-the-art parallel supercomputers cannot match the processing power of the brain's massive neural structure. Nevertheless, the digital computer is undeniably the success story of the century. Its computational finesse has manipulated data for thousands of applications, ranging from household financial planning to missile guidance. The computer, however, can do what a human cannot, fast numerical computation.

Yet, as machines begin to require more human characteristics, such as vision and natural language processing, it has become necessary to re-evaluate expectations. Number crunching might temporarily have to take a back seat. All of a sudden computers are needed to solve problems for which the current systems are unsuited. The world desires another type of machine that can solve problems like a human.

This enormous opportunity has been seized by Artificial Intelligence researchers. Historically, AI has sought to understand how the human brain functions in order to model it "artificially." It is exactly this expertise which brings AI personnel to the forefront of robotics vision research. How to detect and understand sensory stimuli has become a paramount issue. The success of current and future robotic systems, like the FTS, depends on it.

Color as an Additional Parameter

Plans for the FTS vision system include its ability to track and recognize objects as well as visually inspecting the success or failure of robot and human operations. AI vision will be the "eyes" of the FTS and, like real eyes, the recognition of objects can be achieved through edge detection. (Marr, 1982)

An edge detector is used to identify the outline, or contour, of an image. The resulting "edged" image reduces the amount of less significant data while extracting the most significant features. The new data set is in a more compact format allowing for swifter image analysis.

Typically, model-matching has been a common technique utilized for identifying an object. This method employs a knowledge base which has previously stored the edged data for many known objects. The data might include features such as length, number, and orientation of edge elements, etc. By producing the edged data for an object and comparing it to the knowledge base, similarities and differences can be found. Given that enough features are determined to "match," the object can be identified with reasonable certainty.

It is desirable for the robot to have an "intelligent" means for reducing the search space. If a single unknown object must be compared against every object in the knowledge base to determine the best possible match, then as the number of objects in the knowledge base increases, the comparison becomes excessively time-consuming and expensive. It is, therefore, desirable to expedite the search process by reducing the search space.

Striving toward this goal, one promising avenue of research involves adding color information to the knowledge base. The requirements for a color system increases the complexity of the hardware when compared to a gray-level system. However, the value of reducing the search space far outweighs this cost.

The proposed system requires color-coding the objects the system will view. In the case of the FTS, this could mean coloring an ORU (Orbital Replacement Unit) red, a thermal utility connector yellow, etc., or specific parts of the objects might be colored or given certain combinations of colors. This new information for each object provides an additional parameter which the model-matcher can use to better discriminate between objects.

Effectively, color becomes the primary feature used for classification. The vision system will now use its apriori knowledge of what it is seeking to converge upon a solution more quickly. If a red object is detected, the system will search only through known red objects for a match. The multitude of other colored objects are ignored and can be immediately eliminated as likely possibilities.

Image Algebra

AI researchers who seek to design vision systems are necessarily reliant on the quality of low-level information that can be provided. This information is achieved by analyzing unknown input data in terms of known facts contained in the knowledge base. The correctness of the previously determined facts in the knowledge base directly affects the higher-level reasoning of a vision system as does the precision of the input data. Conclusions based on relations between the input data and the knowledge base are key to image understanding.

Image Algebra offers a standardized method for storing and manipulating image information in a knowledge base. The strength of this combination lies in its uniformity: the image is always stored and manipulated in the same way,..*algebraically*. Additionally, the characteristics of Image Algebra provide an excellent means for performing edge detection on an image. In keeping with the ultimate goal of image understanding, edge detection plays an important role.

Most importantly, color images can be processed to find color edges. Compared to gray-level, color edge detection offers a new level of information which can be exploited by a vision system. Overall, this field of study is constantly a source of new techniques for dealing with particularly vexing problems.

Defining an Image using Image Algebra

Digital images are typically represented in discrete form as a 2-dimensional array of pixel values. Each pixel corresponds to an (x,y) location in the image. When considering a gray-level image, these values correspond to the intensity, or brightness, of each pixel. For example, if an image spanned 0..255 gray-levels, 0 could represent black, while 255 would be the brightest, being white. Extending this representation to accommodate color will be explained momentarily.

Image Algebra uses a polynomial of two variables, 'x' and 'y', to define a 2-dimensional image. Similarly, a polynomial of three variables, 'x', 'y', and 'z', is used when defining a 3-dimensional image. However, only 2-dimensional images are considered in this paper.

Constructing a polynomial for an image requires only spatial domain information. The complete polynomial will consist of N x M terms, corresponding to the number of rows and columns in the image. The exponents of the 'x' and 'y' variables correspond to the (x,y) location in the image. The coefficient of each term is representative of the pixel value at the location designated by the exponents. This could be either an intensity/brightness or color value depending on whether a gray-level or color image is being defined.

For simplicity, consider a binary image as seen in Fig. 1(a). A binary image limits the pixel values to being either black (1), or white (0). As shown, the grid defines the (x,y) location of the pixels, while the contents of the grid, a black mark or nothing, is symbolic of the value. For this example only, it is assumed that the object is black on a white background.

Fig. 1(b) is a blow-up of the lower left-hand corner of Fig. 1(a). Here the (x,y) coordinates have been added for reference. This corner, (0,0), is considered to be the origin of the image.

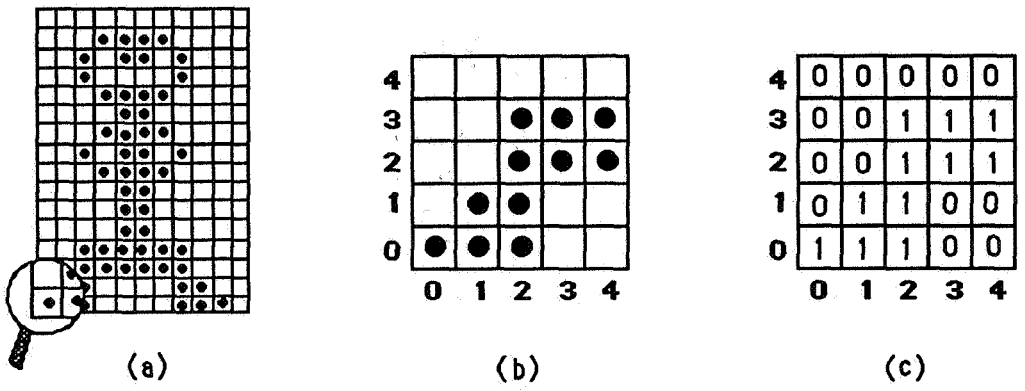


Fig. 1. An example of digital image representation. (a) A digital image. (b) A blow-up of the lower left-hand corner of the image with (x,y) coordinates. (c) Binary pixel values for the lower left-hand corner.

Now, consider the information contained in Fig. 1(c) showing the pixel values. Remember that when constructing the image polynomial, the pixel values are coefficients. Observing Fig. 1(c), it is apparent that many of the terms will be 0. Concentrating on the non-zero terms, the image can be said to be a collection of the following points.

[(0,0), (1,0), (1,1), (2,0), (2,1), (2,2), (2,3), (3,2), (3,3), (4,2), (4,3)]

Using these (x,y) values as exponents, a polynomial expression can be created. It is understood that the coefficient of each term is 1.

$$X^0Y^0 + X^1Y^0 + X^1Y^1 + X^2Y^0 + X^2Y^1 + X^2Y^2 + X^2Y^3 + X^3Y^2 + X^3Y^3 + X^4Y^2 + X^4Y^3$$

Notice how each black pixel is represented according to its (x,y) location. The white pixels have a coefficient of 0 and are not shown because they reduce to zero.

Finally, the polynomial is reduced to a more readable form.

$$1 + X + XY + X^2 + X^2Y + X^2Y^2 + X^2Y^3 + X^3Y^2 + X^3Y^3 + X^4Y^2 + X^4Y^3$$

Defining a Color Image

When using color, each pixel in the image is assigned a value corresponding to its particular color. Working with eight colors, Fig. 2 below shows the color assignments. Notice that red, green, and blue are assigned to the powers of 2 ($2^0 = 1$, $2^1 = 2$, and $2^2 = 4$). This will become important later.

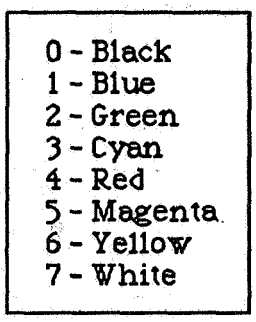


Fig. 2. Color Assignments

Fig. 3 illustrates a region of a color image containing the colors red, green, and magenta on a black background. Again, for simplicity, only a small region near the origin of the image is displayed. This keeps the exponents small and more readable. In reality, exponents assume very large values due to the considerable size of most images. For example, a 512 x 512 image would require exponents 0..511.

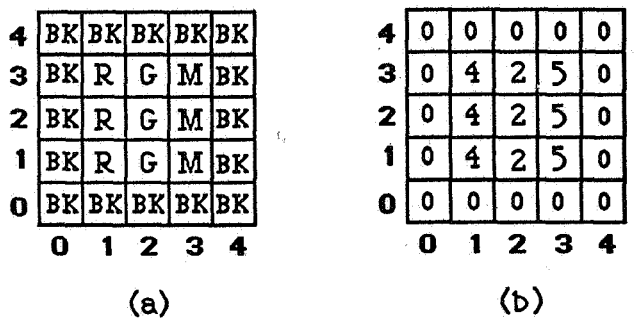


Fig. 3. An example color image region. (a) A red, green, and magenta image. (b) Color pixel value assignments using definitions from Fig. 2.

Constructing the polynomial for a color image is very similar to that of the binary case discussed earlier. For example, the image region in Fig. 3 is comprised of the following set of points.

{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)}

In the binary illustration of Fig. 1, the coefficients were either 0 or 1. Working with color, the pixel values can be 0..7. The color coefficients, representing pixel value, therefore assume the range 0..7. Writing the reduced polynomial with the proper color coefficients yields the polynomial below.

$$4XY + 4XY^2 + 4XY^3 + 2X^2Y + 2X^2Y^2 + 2X^2Y^3 + 5X^3Y + 5X^3Y^2 + 5X^3Y^3$$

Image Operations

The operations applied to image polynomials may be initially conceptualized as being the addition and multiplication of polynomials according to the standard arithmetic methods.

Consider multiplying the two small images in Fig. 4. The polynomial expressions for the images are also shown.

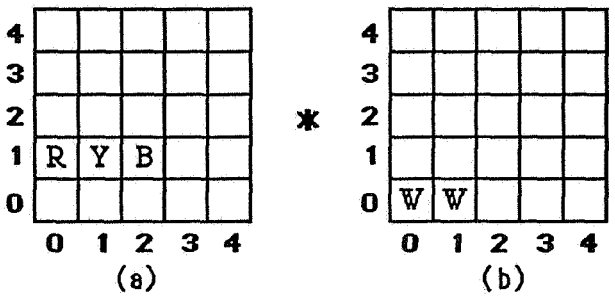


Fig. 4. Color Images (a) $4Y + 6XY + X^2Y$ (b) $7 + 7X$

Using standard algebraic technique, the two polynomials multiplied together yield the following result.

$$(4Y + 6XY + X^2Y) * (7 + 7X) = 28Y + 70XY + 49X^2Y + 7X^3Y$$

Clearly, the large coefficients in the resulting equation present a problem. The colors are defined only in the range 0..7, but this equation contains coefficients as large as 70, and no color assignments were created for anything larger than 7.

Redefining the addition and multiplication operations on the color coefficients {0,1,..,7} so that the set is mathematically closed, solves this problem. Closure requires both operations to produce a result which is within the set's limits. In this manner, the coefficients generated will always remain within the range of color assignments.

Fig. 5 shows the new definitions for addition (+) and multiplication (•). The addition table has been produced using bitwise OR while the multiplication table has been produced using bitwise AND (Qian & Bhattacharya, 1991). It should be noted that addition is used to combine similar terms resulting from the multiplication.

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	1	3	3	5	5	7	7
2	2	3	2	3	6	7	6	7
3	3	3	3	3	7	7	7	7
4	4	5	6	7	4	5	6	7
5	5	5	7	7	5	5	7	7
6	6	7	6	7	6	7	6	7
7	7	7	7	7	7	7	7	7

(a)

•	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	1
2	0	0	2	2	0	0	2	2
3	0	1	2	3	0	1	2	3
4	0	0	0	0	4	4	4	4
5	0	1	0	1	4	5	4	5
6	0	0	2	2	4	4	6	6
7	0	1	2	3	4	5	6	7

(b)

Fig. 5. Image Algebra Operations (a) Addition (b) Multiplication

Now that the set addition and multiplication are being used, the polynomial operation considered before becomes the following. As expected, the coefficients fall within the defined range.

$$(4Y + 6XY + X^2Y) * (7 + 7X) = 4Y + 6XY + 7X^2Y + X^3Y$$

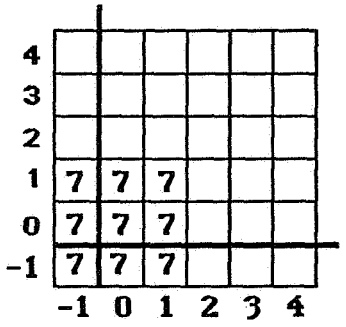
Especially relevant, is that the operations maintain the integrity of the color combinations. For example, blue and red are known to yield magenta (1 + 4 = 5), green and red yield yellow (2 + 4 = 6), etc. Also, a color added to itself, remains the same (3 + 3 = 3).

Color Edge Detection with Image Algebra

In the previous example of Fig. 4, it was demonstrated how two polynomials could be multiplied together. In essence, the first image/polynomial is very large. Images are usually at least 512 x 512 pixels. The second image/polynomial is usually only a few pixels. This small polynomial is often distinguished as being an *operator* or *image operator*. When the operator is created specifically for the purpose of edge detection, it is often referred to as an *edge detector operator*, or simply an *edge detector*.

Edge detection is a result of constructing the pixels in the image operator in such a way that an edge/contour is produced when it is multiplied with the original image. The image operator usually includes the term '1' (X^0Y^0), plus other terms which will shift the image 1-pixel away from its original position in various directions. For example, X would shift the image 1-pixel in the X direction, while XY would shift the image 1-pixel in the XY direction. This process smears the image. Next, the original is subtracted from the smeared image to remove the interior of the region. This leaves the desired outline of the object. The subtraction is performed using a bitwise XOR.

Consider the edge detector shown in Fig. 6 which contains coefficients corresponding to white(7) (Qian & Bhattacharya, 1991). This operator consists of a 3 x 3 square centered around the origin. The negative exponents assure that every shift in the positive direction, is offset by another in the negative direction. In this manner, all edges regardless of orientation, will be found.



$$7 + 7X + 7XY + 7Y + 7X^{-1}Y + 7X^{-1} + 7X^{-1}Y^{-1} + 7Y^{-1} + 7XY^{-1}$$

Fig. 6. Image Operator for Color Edge Detection

An example of this operator being applied to an image is shown in Fig. 7.

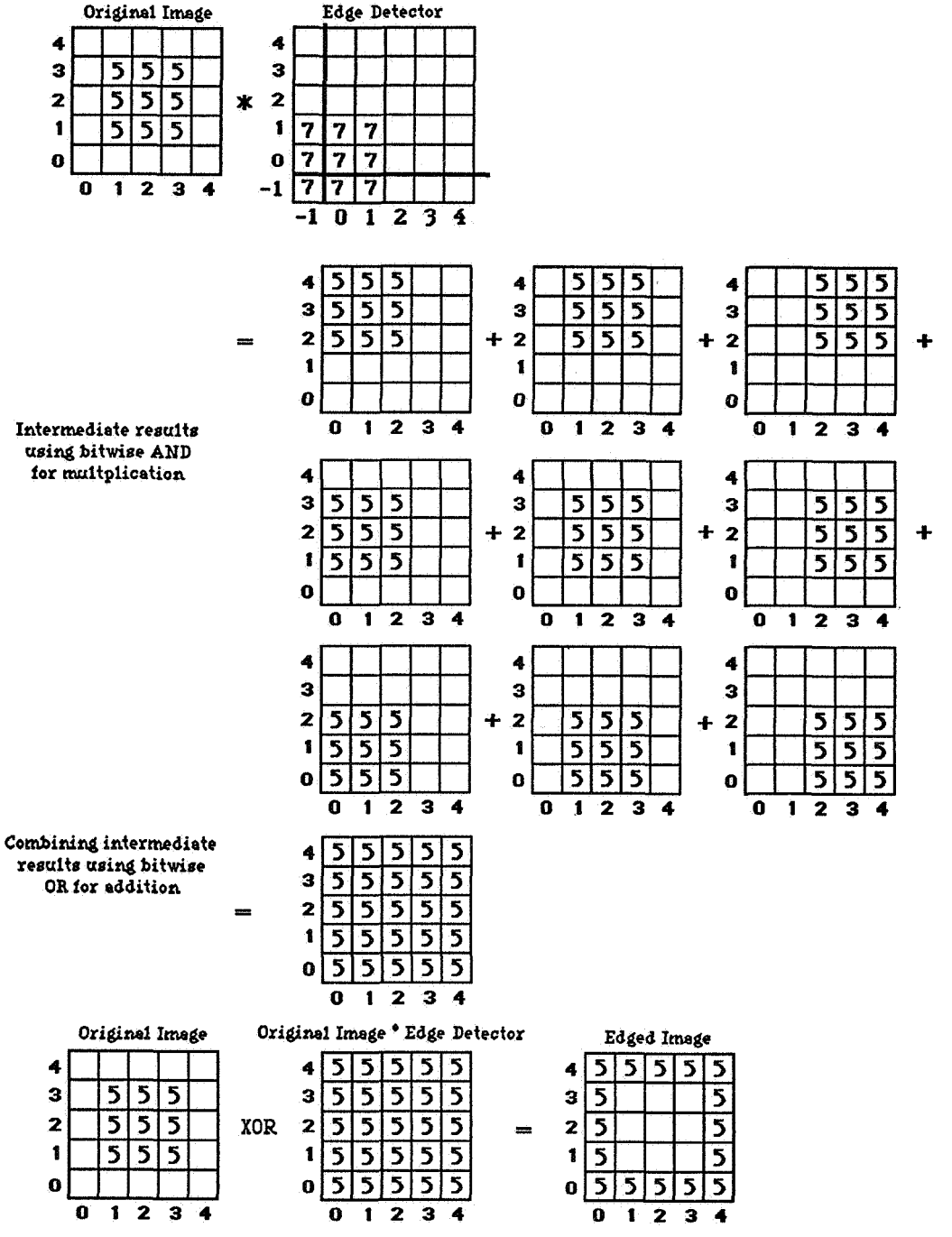


Fig. 7. An example of the color edge detector being applied to a color image.

The strength of this approach is that the coefficients of the edge operator can be changed so that only edges of a certain color are found. The advantage this offers is evident when again considering the requirements for a vision system's knowledge base. Edge detection reduces the data while maintaining the most important image features. When only specific color information is extracted, even more unwanted data is able to be discarded, while more relevant data is brought to the attention of the AI system.

Consider the color assignments in Fig. 8.

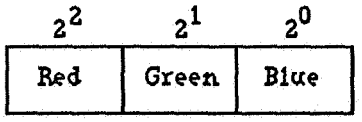


Fig. 8. Primary Color Assignments

All colors that can be displayed on a monitor are created by varying intensity levels of the three electronic primary components, red, green, and blue. Certainly, no system has the capacity to display all potential colors. The number of colors that can be displayed is dependent on the number of bits representing each primary color.

Using 3 bits, 1 for each primary color, 8 unique colors can be generated. Because the 3 primary colors were assigned to the powers of 2, the 8 resulting combinations correspond with those in Fig. 2. This is due to the fact that the particular 8 colors are formed from simple combinations, none of the colors (black), all of the colors (white), one of the colors (red, green, or blue), or two of the colors (magenta, cyan, or yellow).

The binary representation of the eight colors can be seen in Fig. 9. A '1' means that the primary color in that position is at full intensity. A '0' means that that color is not present. For example, cyan is composed of green and blue, but not red. This is shown as a '1' in both the green and blue bits, but not the bit which corresponds to red.

000 - Black
001 - Blue
010 - Green
011 - Cyan
100 - Red
101 - Magenta
110 - Yellow
111 - White

Fig. 9. Binary Color Assignments

Consider again the example presented in Fig. 4. This image was represented by the polynomial below.

$$4Y + 6XY + X^2Y$$

The coefficients represent the color assignments. Presented here in integer form, the same equation could also be written in binary form as shown below. 4 is written as 100_2 in binary, 6 as 110_2, etc.

$$100_2Y + 110_2XY + 001_2X^2Y$$

Going back to the edge detector operator, it is seen that the coefficients are all 7, or 111_2. The result of using 7 for the coefficients is that all edges are found in all three color planes, red, green, and blue. If the coefficient were either red (4), green (2), or blue (1), only the edges of that color would be found. If, on the other hand, a magenta (5 or 101_2) was used, red and blue edges would be found, but not green. Magenta's binary representation does not have a '1' in the middle/green bit.

Because color cannot be reproduced for this publication, the following examples are provided to demonstrate the edge operator. First, consider the image in Fig. 10. This example shows an image containing seven shapes. The background is assumed to be black (it is not shaded for readability purposes). Assume the image is multiplied by the edge detector operator. The coefficients of the operator are all '7', 111_2 in binary. Therefore, all edges of all colors will be found because there is a '1' at each primary color bit. All edges will be the color of the object they correspond with, i.e. the red object will produce a red edge, the yellow object a yellow edge, etc.

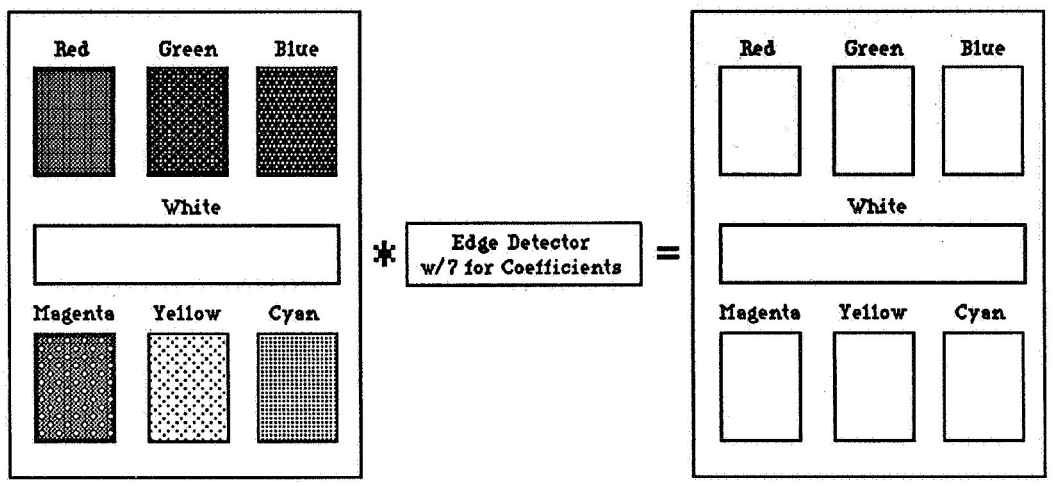


Fig. 10. Example of Edge Detection for All Colors

Now consider the same image, but multiplied by the edge detector with coefficients of 4, red. In this case, only objects which contain red, regardless of other colors, will be detected. As seen in Fig. 11, the red, white, magenta, and yellow objects have been found. These colors all have a '1' in the red color bit. In this case, the edges will all be red, not the color of the object.

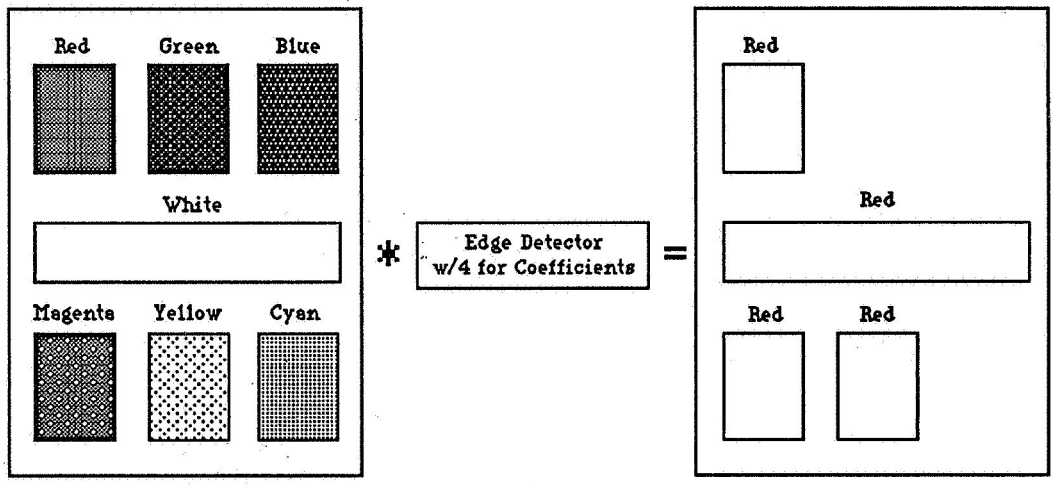


Fig. 11. Example of Red Edge Detection

So far, this publication has only discussed color image operators that have been applied to objects on a black background. Working with colors on a non-black background is not always as straightforward. Consider the case shown in Fig. 12 where a red object is on a blue background. When the edge detector is applied, the result is a blue colored edge inside a red colored edge. At first glance, the edges might seem reversed from what is expected (blue inside red, instead of red inside blue). However, because the edge of an object is comprised of those pixels immediately external to its boundary, the corresponding red and blue edges are indeed in the right place.

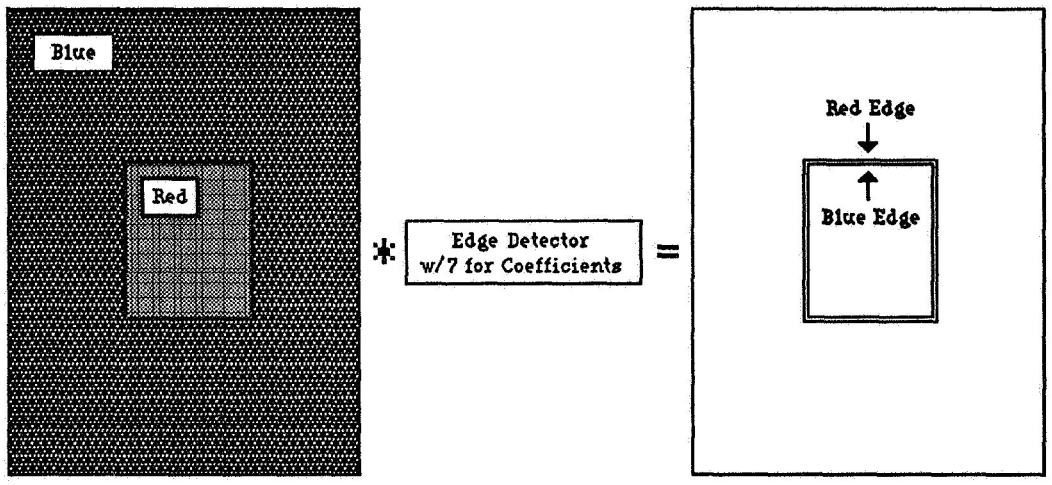


Fig. 12. Edge Detection Performed on a Red Object with a Blue Background

As mentioned earlier, working on a non-black background is a considerably more complicated problem. Two colors adjacent to one another do not always produce edges whose colors can be anticipated. The mixture of red and blue produces the expected red and blue edges only because their bit representations do not have any bits in common, red being 100_2 and blue 001_2. If two colors are adjacent to one another *and* have bits in common, the shared bit will be canceled out during the algebraic operations. For example, yellow (110_2) and magenta (101_2) share the red bit, so the edges produced will be green (010_2) and blue (001_2) because the red bit is canceled. This feature of color Image Algebra will continue to be a topic for future research.

Robotics Application

Vision capabilities for robotic systems are critical to establishing autonomous decision-making. For example, NASA engineers expect to supervise the FTS from Earth as well as from space. It is, therefore, critical that the robot be able to respond to as many situations as possible without requiring human assistance. Transmitting communication at every turn would quickly become a burden for both the robot and the human.

The grand scheme for incorporating color Image Algebra into a robotic system like the FTS includes color-coding the objects the robot is expected to interact with. If the objects were to be the same color as the space station truss, or black like the dark space background, little contrast would be available to a vision system. On the other hand, by coloring the objects, not only will they stand-out against the background, but they will be more easily recognized by the knowledge base because of the additional color information.

The technique described throughout this paper obviously lends itself to this robotic application. The Image Algebra approach offers both speed and accuracy. Additionally, the knowledge base of object data can be built up to accommodate classes of objects corresponding with object-oriented design. Similarly, the number of colors can also be increased so that there is a 1-to-1 correspondence between how the image is physically stored and how it is algebraically represented.

CONCLUSIONS

Image Algebra and color combine as an innovative approach to fulfilling image processing requirements for an AI-based vision system. The algebraic approach provides a solid foundation for image representation as well as a method for edge detection. Color has also proven itself as a useful parameter for discriminating between objects in a scene. The ease and flexibility of this approach makes it a valuable image analysis technique for robotic space applications.

ACKNOWLEDGEMENTS

The author would like to acknowledge the work of Dr. Prabir Bhattacharya at the University of Nebraska, Lincoln whose work (NAG 5-1382) formed the foundation for this paper. Thanks to Tom Winkert and Colleen Hartman of Code 735.1 NASA/Goddard Space Flight Center for their good humor throughout this project. Also, thanks to Bob Cooke of Cooke Publications, Ltd. for a copy of MathWriter, a scientific word processor for the Macintosh.

REFERENCES

Marr, D. (1982) *Vision*, Freeman, San Francisco, Calif.

Qian, Kai, & Bhattacharya, P. (1990, June) Binary Image Processing By Polynomial Approach, *Pattern Recognition Letters 11*, 395-403.

Qian, Kai, & Bhattacharya, P. (1991) An Algebraic System for Operating On Gray, Color and 3-D Images. Submitted for Publication.